Visualization of Radiation For Optimized Design of Automotive Antennas

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Engineering with distinction in the College of Electrical and Computer Engineering

By

Ryan A. Tokola

****

The Ohio State University

Winter 2010

BS Committee

Dr. Joel Johnson

Dr. Eric Walton

Advisors Approval

_____

Advisor

College of Engineering

ACKNOWLEDGEMENTS

Thanks to Dr. Joel Johnson for advising me on this project, Dr. John Young, and Eugene Lee for their helpful advice, Wladimiro Villarroel and the engineers at AGC America for their unflagging support of this project, and Dr. Edward Newman for allowing the use of his excellent ESP5 software. Extra thanks to Dr. Eric Walton for his ceaseless support, encouragement, and advice.

Table of Contents

Table of Figures

1. Introduction

1.1 Background

The number of wireless consumer devices is rapidly increasing, resulting in a proportional increase in the number of antennas on modern automobiles. Cars may be required to receive signals from sources including AM/FM radio, satellite radio, cellular telephone, satellite telephone, GPS, the intelligent highway system, remote keyless entry, Bluetooth, VHF/UHF television, and collision-avoidance radar. This covers a very broad frequency range, from 0.5 MHz to 5GHz. Rather than design and implement a separate antenna for each function, the use of multifunctional antennas is becoming a popular focus of contemporary research. Multifunctional antennas are capable of operation at multiple frequencies, without the cost and space problems associated with individual antennas for each specific frequency range.

On a vehicle, however, a single multifunctional antenna is insufficient for two reasons. First, no antenna, even if it is multifunctional, will meet gain, phase, and radiation pattern requirements for the entire frequency range under consideration. Second, even if such an ideal antenna existed, practical considerations would prevent it from acceptable operation under all conditions. Problems such as multipath interference, when the radiated signal destructively interferes with its own reflections, will occur with single antenna systems. Also, conformal antennas at higher frequencies will experience vehicle blockage.

It is logical, then, to strategically locate a small network of multifunctional antennas on the vehicle body and combine the signals. The signals from these antennas, once intelligently mixed or selected, yield expanded frequency and spatial coverage, and can mitigate the effects of multipath interference and blockage. It is possible to place antennas nearly anywhere on a

vehicle, but on-glass antennas are an increasingly attractive option to automobile manufacturers, as they are low-cost, unobtrusive, and easy to manufacture.

Although many techniques for accurately simulating the electromagnetic characteristics of antennas exist, it is difficult to design a good antenna system. Antennas are frequently designed through an ad-hoc combination of experience, intuition, and guesswork. Designing antennas for multiple frequency ranges and diversity applications is even more complicated. There exists a need for new technologies and methodologies to supplement traditional antenna design techniques.

This project was originally conceived as an application that would automatically calculate the optimized design of on-glass automotive antenna networks. Users would input vehicle geometry, select initial antenna geometries, and an optimization algorithm would modify the antenna geometry until a satisfactory design had been achieved. It became apparent, however, that there were many difficulties with such an automated process. How would an optimization cost function be designed? How could designers verify that the cost function, or even the specific optimization algorithm, is suitable for the specific vehicle under consideration? How could it be know if the optimization was stuck in a small local minimum, with a much better solution elsewhere? How could the designer verify that the solution provided by the optimization algorithm could tolerate small deviations in geometry? The only way these questions can be answered is if engineers are more involved in design process. This project has become more focused as a result, with a new emphasis on enhancing designer's understanding of antenna performance.

1.2 The Benefits Visualizing Radiation Data

Designing antenna systems necessarily involves working with large amounts of data. Whether generated by numerical simulation or empirical measurement, these data are frequently many-dimensional and rarely smooth and intuitive. To fully appreciate the significance of these sets of data, it is often helpful to generate various plots to visually see what is happening. These plots are useful for any design process. Given calculated or measured radiation pattern data, an antenna designer may use plots for several purposes:

- Most obviously, it is important to assess the effectiveness of a specific antenna design. A plot of an antenna's radiation pattern can quickly show if it meets design criteria, and where it may need improvement. A plot can reveal patterns that may not be obvious from raw numbers, and these patterns can suggest ways of improving the design. Here, polar plots of an antenna's radiation patterns will be used.

- It is often desirable to understand how the behavior of an antenna will change when a geometric parameter has changed. A set of simulations, sweeping the parameter over a certain range, can generate data that demonstrate the effect of changing the parameter. In this case, rectangular plots with azimuthal angle on one axis and values of the swept parameter on the other axis efficiently express the relationship between the parameter values and the radiation patterns of the antenna.

- To appreciate how two geometric parameters interact, it may be useful to see the effect of both parameters on a single plot. To accomplish this, a rectangular plot can be constructed that has one swept parameter on each axis. Of course, this leaves a single

data point for each 360˚ radiation pattern. Some function must be used to convert the radiation pattern into a scalar value.

- The application of optimization algorithms to electromagnetic design is an active area of research. Classical gradient-based optimization methods and newer global optimization strategies are being used to automate design processes and arrive at solutions that might not have been found "by hand." See the references at the end of this chapter.
    - Optimization algorithms operate by evaluating the value of cost functions, which are single-valued assessments of the effectiveness of a design. The careful design of a cost function is critical to the successful application of an optimization algorithm. The cost function must take into account all design criteria, and balance them in an effective way. Also, the cost function solution space must be relatively smooth for optimization algorithms to work effectively. Plots of cost function values can be used to design, analyze, and refine the cost functions.
    - With vigorous research being dedicated to many radically different optimization algorithms, it is not always obvious which approach is appropriate for any given application. Plots of antenna radiation patterns can assist with the decision. Antenna simulations performed over several swept geometric parameters can help indicate how radiation patterns change with varying antenna geometry, and can show which types of optimization would be effective. For example, radiation patterns that vary smoothly across swept parameters could work best with local gradient-based algorithms, whereas radiation patterns with large numbers of local minima may

require global optimization techniques such as genetic algorithms or particle swarm optimization.

  - If an optimization algorithm that finds local minima is to be used, plots with swept parameters can be used to find promising neighborhoods for optimization. The plots can be used to find a good starting point, and then an optimization algorithm can refine the geometric parameters.

- Plots are also useful for comparing data. Plotting the results of the simulation of two alternative antenna configurations allow easy comparison of the relative advantages of each design. Also, the accuracy of simulations can be checked by comparing plots of simulation data and actual test measurements.

In all of these cases, an "antenna" may be a single antenna or a set of antennas. If multiple antennas are being considered, their signals are assumed to have been selected or combined in some way. Also, a "radiation pattern" may be one of several antenna characteristics, such as vertical or horizontal polarization, a circular polarization, or any combination of these.

1.3 About VAAR

Visualization of Automobile Antenna Radiation (VAAR) is a MATLAB-based tool created to help design networks of on-glass automobile antennas. It is used to simulate the radiation patterns of on-glass antennas and view and compare the resulting data using flexible, user-defined functions.

VAAR has two main functions, which have associated Graphical User Interfaces (GUIs):

- Define antenna geometries, set up parameter sweeps, and generate simulations of the radiation patterns of the antennas with all combinations of selected parameters. This is accomplished with **modelGUI**.
- Visually inspect the resulting data, with the ability to compare data sets side by side. Data is loaded in **vizSetupGUI** and the resulting plots are shown in **vizPlotGUI**. With minor improvements, it will be possible to load data from actual measurements of antenna radiation patterns, which will allow easy comparison of theoretical and real world data.

VAAR's visualization tools can assist with:

- Selection of an optimization algorithm.
- Development of a cost function.
- Identification of good neighborhoods for local optimization.
- Comparison of theoretical and measured data.
- Assessment of optimization performance.

The overall antenna design process may look something like:

1) Import automobile geometry
2) Define a starting set of antennas
3) Simulate response with parameter variations
4) Use visualization tools to refine antenna design via traditional techniques

or -

4) Use an optimization algorithm
   a) Use visualization tools to identify appropriate optimization algorithm

b) Develop / refine cost function

c) Identify which parameters will be varied by optimization

d) If local optimization is selected, locate good neighborhoods for starting points

e) Apply optimization algorithm

5) Compare final simulated design with measured prototype response

It is intended to be flexible and expandable. It should not be difficult to add components to enhance its functionality. For example, although it is possible to use VAAR as a way of setting up an optimization algorithm, it may be preferable to incorporate some components of VAAR into an optimization system. One advantage of using VAAR for the optimization is that every iteration's data will already be in a format that can be used with VAAR's visualization tools. Data may be visually compared to the results of parameter sweeps. This provides the designer with quick and easy feedback about the performance of the optimization.

MATLAB is required to convert measured data to a format readable by VAAR. Therefore, VAAR is not currently available as a standalone application. VAAR requires MATLAB version 2009a or later. No MATLAB toolboxes are required.

The simulation engine is the ESP5 Electromagnetic Surface Patch (Method of Moments) code, created by Edward H. Newman. Further information about ESP5 can be found at:

http://ckm.osu.edu/sitetool/sites/pdfs/eslpublic/research/esp5_sum5_4.pdf

VAAR uses wire meshes to define automobile geometry. Although a function that creates simple parametrically defined vehicle meshes is included with VAAR, users will be expected to provide vehicle mesh files that correspond to their current projects. The format that VAAR uses

for vehicle mesh files is very simple, and converting existing geometry files to VAAR's format is not expected to be difficult.

VAAR has been developed without real world testing to verify the accuracy of simulation results. This is partially justified by the known quality of ESP5, which performs that actual simulations. However, it is important to be aware of that ESP5 can only be as accurate as the information given to it. The vehicle meshes used in the development of VAAR were extremely simple, which allowed for short simulation times. Vehicle meshes used in practice will likely be much more dense, and simulation times increase dramatically with greater wire mesh density. Preliminary testing has indicated that simulation results made with vehicle meshes of increasing density do not converge until the simulations take a prohibitive amount of time.

VAAR is an Ohio State University research project, and as such it cannot be warranteed by OSU beyond its initial release. The final raw computer code (written in MATLAB) will be delivered with this project. It is hoped that this User's Guide and documentation within VAAR's code will be sufficient to allow AGC to correct, improve, and expand the software. The **Future Improvements** section on page 46 discusses several ways that VAAR may be improved.

VAAR uses distance in meters and angles in degrees. Text that appears in the GUIs will appear in bold in the User's Guide.

2. Initial Setup

modelGUI

The m-file **modelGUI** is used to prepare and run parameter sweeps that vary antenna geometry and/or simulation frequency. **modelGUI** is shown with a completely defined sweep in Figure 1.

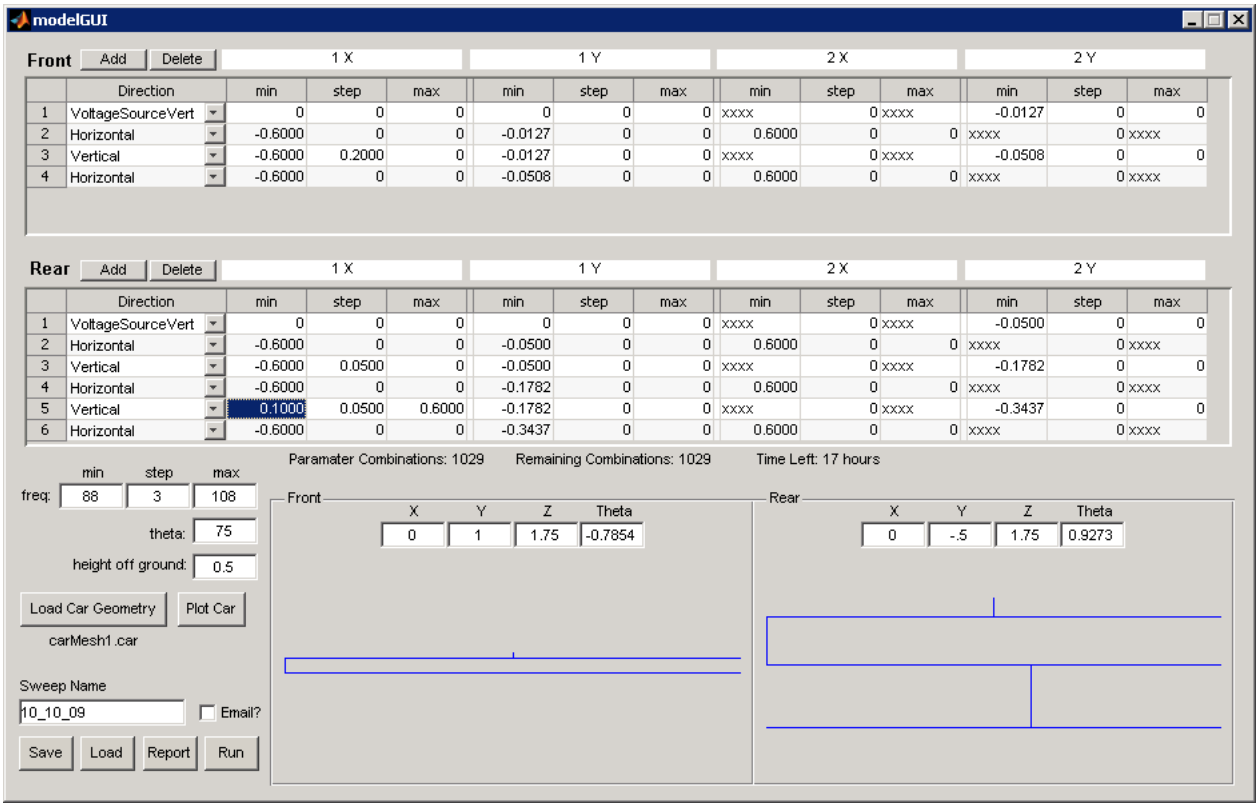


Figure 1: modelGUI

modelGUI is typically used in the following sequence:

- 2.1 Define Antenna Geometries
- 2.2 Load Vehicle Wire Mesh Geometry
- 2.3 Locate The Antennas On The Vehicle
- 2.4 Set Simulation Parameters
- 2.5 Name and Save
- 2.6 Generate a Report
- 2.7 Run the Simulation

2.1 Define Antenna Geometries

Antennas are defined in **modelGUI** using the **Front** and **Rear** antenna definition tables. It is currently necessary for both antennas to be defined with at least one swept parameter (see section 8. Future Improvements 1 and 2). Although it is common for antennas to be located on the front and rear windshields, the designations "front" and "rear" are arbitrary—antennas can be placed anywhere on a vehicle model. An antenna definition table is shown in Figure 2.

Rear | Add | Delete

		1 X			1 Y			2 X			2 Y		
	Direction	min	step	max	min	step	max	min	step	max	min	step	max
1	VoltageSourceVert	0	0	0	0	0	0	xxxx	0	xxxx	-0.0500	0	0
2	Horizontal	-0.6000	0	0	-0.0500	0	0	0.6000	0	0	xxxx	0	xxxx
3	Vertical	-0.6000	0.0500	0	-0.0500	0	0	xxxx	0	xxxx	-0.1782	0	0
4	Horizontal	-0.6000	0	0	-0.1782	0	0	0.6000	0	0	xxxx	0	xxxx
5	Vertical	0.1000	0.0500	0.6000	-0.1782	0	0	xxxx	0	xxxx	-0.3437	0	0
6	Horizontal	-0.6000	0	0	-0.3437	0	0	0.6000	0	0	xxxx	0	xxxx

Figure 2: Antenna Definition Table

Antennas are composed of a set of wires. Each numbered row of the antenna definition table corresponds to one wire. The column sections labeled **1X**, **1Y**, **2X**, and **2Y** correspond to the x- and y-coordinates of the first and second wire endpoints. Because the coordinate system used to define antennas is independent of the vehicle's coordinate system there is no need for a z-coordinate. Currently, antennas can only be defined on a flat planar surface (see section 8. Future Improvements 3).

Any endpoint of any wire can be varied in either the x- or the y-direction in a parameter sweep. If the **step** column of an endpoint is set to 0, the endpoint will not be swept on that axis, and the value in the **min** column will always be used when defining the antenna. If the **step** column contains a positive nonzero value the endpoint will be defined on all values from **min : step : max** (using MATLAB notation). All endpoints must be defined such that **min** < **max** and **step** ≥ 0. Furthermore, horizontal wires require **1X** < **2X** for all values of **1X** and **2X**, and vertical wires require **1Y** < **2Y** for all values of **1Y** and **2Y**. If a **step** will be used, it is necessary to have the **max** value set greater than the **min** value prior to entering the **step**.

When creating a new wire, first choose the wire type from the pull-down menu in the **Direction** column. Select either **VoltageSourceHoriz** or **VoltageSourceVert** for the first wire (see section 8. Future Improvements 5). **VoltageSource** wire types have a voltage source attached at the first point of the wire. It is this voltage source that drives the radiation pattern that ESP5 will calculate. Because the first endpoint of the first wire typically attaches to the vehicle wire mesh, it is usually located at (0,0) so the antenna coordinate system can easily be mapped onto the vehicle coordinate system. (See section 2.3 Locate The Antennas On The Vehicle, page 19). When a **horizontal** wire has been created, the **2Y** column will contain `xxxx | 0 | xxxx`. This indicates that, as a **horizontal** wire, it is unnecessary to define a second y-

coordinate. The same is true for the **2X** column when **vertical** wires are created. No changes to these columns should be made.

After the first wire, all subsequent wires must be either **horizontal** or **vertical**. Wires are allowed to overlap and intersect—a subroutine automatically joins overlapping wires and splits intersecting wires to avoid errors in ESP5 (see section 8. Future Improvements 6 and 7). It is also not necessary for wires to make contact with any other wires. Although the endpoints of two wires may occupy the same point at the beginning or end of a sweep, they should not occupy the same point in the middle of a sweep. For example, if a horizontal wire is located at y = 1, and a vertical wire has an endpoint at (2,1), a second vertical wire should not have an endpoint at (1:0.5:3, 1) because the horizontal wire may be split into a very small wire segment near (2,1). Rounding errors in MATLAB can cause the creation of a very small wire that will cause an error in ESP5 (see section 8. Future Improvements 4).

Impractically long simulation times can easily be caused by an accidental choice of too many parameter combinations. **modelGUI** provides some feedback that can be used to fine-tune the size of parameter sweeps. Information about parameter combinations and simulation time, shown in Figure 3, is displayed below the antenna definition tables. **Parameter Combinations** is the total number of combinations of swept parameters. **Remaining Combinations** dynamically updates during simulation, and can be used to keep track of simulation progress. **Time Left** is an estimate of the remaining simulation time based on an average of previous simulations. **Time Left** is only meant as a general guideline, as simulation times vary with frequency and model complexity (see section 8. Future Improvements).

Figure 3: Sweep Info

It is assumed that variations in the geometry of one antenna will not affect the radiation pattern of the other antennas. Therefore, **Parameter Combinations** is calculated by:

$$\text{Parameter Combinations} = pfreq\left(\prod_i p1_i + \prod_j p2_j\right)$$

where $pfreq$ = # swept frequencies (see section 2.4 Set Simulation Parameters, page 21)

$p1_i$ = # values for the i^{th} swept parameter of antenna 1

$p2_i$ = # values for the j^{th} swept parameter of antenna 2

A plot of the antenna geometry, as shown in Figure 4, will be dynamically updated as the antenna is being defined. If antenna parameters are swept, only the minimum value will be used in the plot (See section 8. Future Improvements 9).

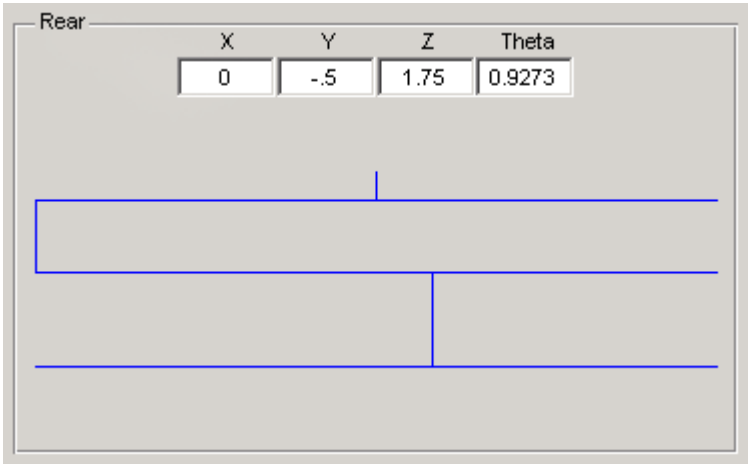


Figure 4: Antenna Location and Plot

2.2 Load Vehicle Wire Mesh Geometry

To load a wire mesh of the vehicle, press the **Load Car Geometry** button, shown in Figure 5.

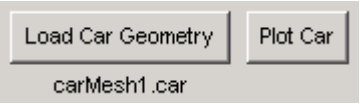


Figure 5: Load Vehicle Mesh / Plot Car and Antenna

It is beyond the scope of this software to assist with the creation of complex vehicle meshes, but `makeCarMesh.m` can be used to create a simple, parametrically defined vehicle model. (See section 5. Vehicle Wire Mesh Geometry Files, page 36).

2.3 Locate The Antennas On The Vehicle

Once antennas have been defined and a vehicle mesh has been loaded, the antennas must be located and oriented on the vehicle. Typically, the first antenna wire will have a voltage source and a first endpoint at location (0,0) in the antenna's coordinate system. The default location of the antennas is with the origin of the antenna's coordinate system at the origin of the vehicle's coordinate system, with the antenna rotated to be in the vehicle's x-z plane (See Figure 6). The values **X**, **Y** and **Z** seen in Figure 4 are used to locate the antenna within the vehicle's coordinate system. The value **Theta** is the antenna's rotation, in radians, about the x-axis. Press the **Plot Car** button, shown in Figure 5, to see a three dimensional plot of the car and antenna, as shown in Figure 7. Although the plot will show the bottom of the vehicle at z = 0, during simulations a mirrored copy of the vehicle (as discussed below) will simulate a ground plane. This hypothetical ground plane is a user-defined distance below the bottom of the vehicle. Note

that no x-axis translation is necessary when the origin of the antenna is placed on the plane of lateral symmetry. (See section 8. Future Improvements 10, 11 and 12). After location changes have been made, it is currently necessary to modify the antenna in its antenna definition table before the location changes are saved. This may be as simple as entering "0.0" into a cell with the current value "0." (See section 8. Future Improvements 13).

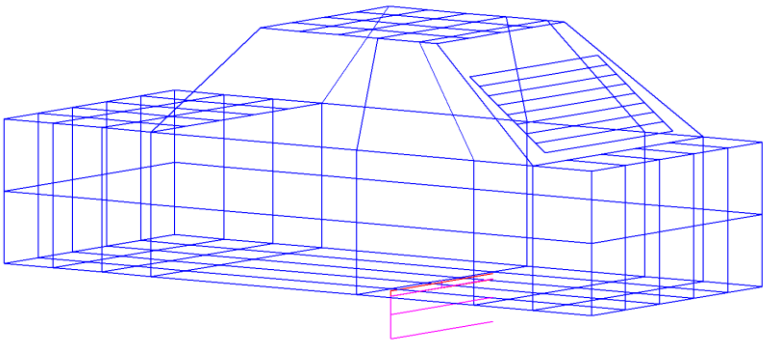


Figure 6: Default Antenna Location

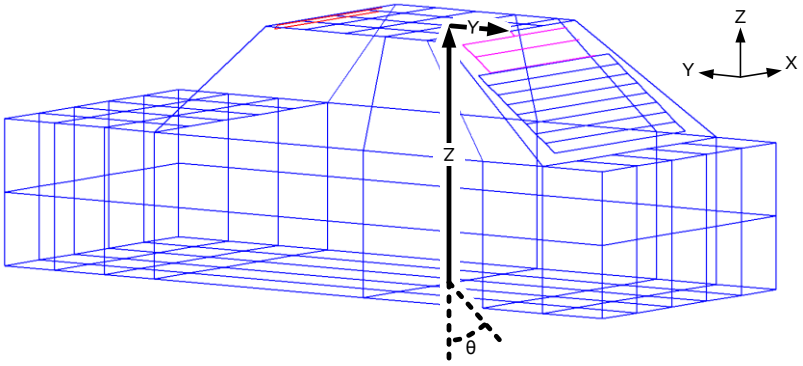


Figure 7: Car With Antennas Properly Located

2.4 Set Simulation Parameters

Frequency sweeps and theta (elevation angle) are defined in the **freq** and **theta** text boxes, as shown in Figure 8 (see section 8. Future Improvements 14). **Height off ground** is the distance from the bottom of the vehicle to the hypothetical ground plane, assuming that the bottom of the vehicle is at z=0 in the vehicle's coordinate system. During creation of the .esp file that is used for simulation, an image of the vehicle is mirrored about the hypothetical ground plane. This is equivalent to modeling the ground plane as a PEC.

Figure 8: Sweep Parameters

2.5 Name and Save

The simulation may be named and saved at any time. The name entered in the **Sweep Name** field, shown in Figure 9, will be used for both the saved simulation information (with the file extension .sim) and the data file that results from the simulation (with the file extension `*.data`), (See section 8. Future Improvements 15). Simulation settings are saved by pressing the **Save** button, and saved settings can be loaded by pressing the **Load** button. When saving, if a *.sim file already exists with the same name, a dialog box will ask if the previous file should be overwritten. Check the **Email?** checkbox to have an email sent when the simulation has complete (see section 6. Email Notification, page 40 and 8. Future Improvements 16).

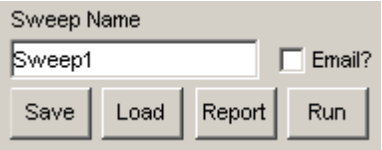


Figure 9: Save/Load/Run Sweep

2.6 Generate a Report

All settings in **vizGUI** can be displayed by pressing the **Report** button. A dialog box, shown in Figure 10, presents three options:

- **Command Window Display** will print the settings to the command widow.
- **Text File** will generate a text file named "[sweep name]_ModelReport.txt." A monospaced font, such as Courier, is recommended for viewing these files. The text file may not conform to the proportions of a standard page (see section 8. Future Improvements 18).
- **Excel Spreadsheet** will generate a spreadsheet named "[sweep name]_ModelReport.xls"

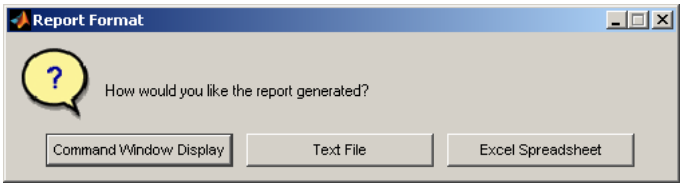


Figure 10: Report Format Dialog

2.7 Run the Simulation

With a vehicle mesh loaded, the antennas defined and located on the vehicle, simulation parameters set, and the simulation named and saved, the simulation may be executed by pressing the Run button.

3. Data Visualization

Visualization GUIs

Two GUIs are used to visualize data after a sweep has been executed. **vizSetupGUI** (see Figure 11) is used to load data and define how the data will be displayed. **vizPlotGUI** is a set of four axes for plots (see Figure 12).

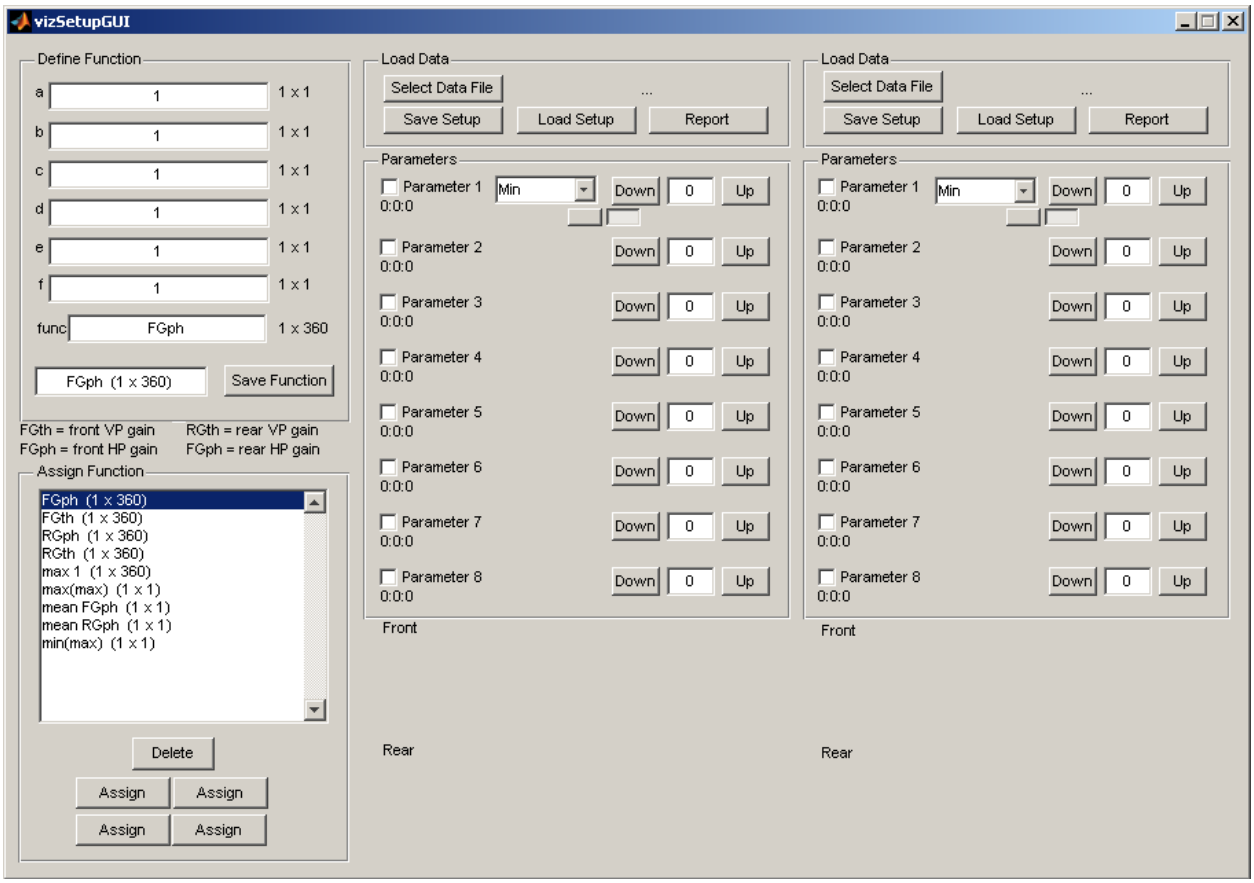


Figure 11: vizSetupGUI

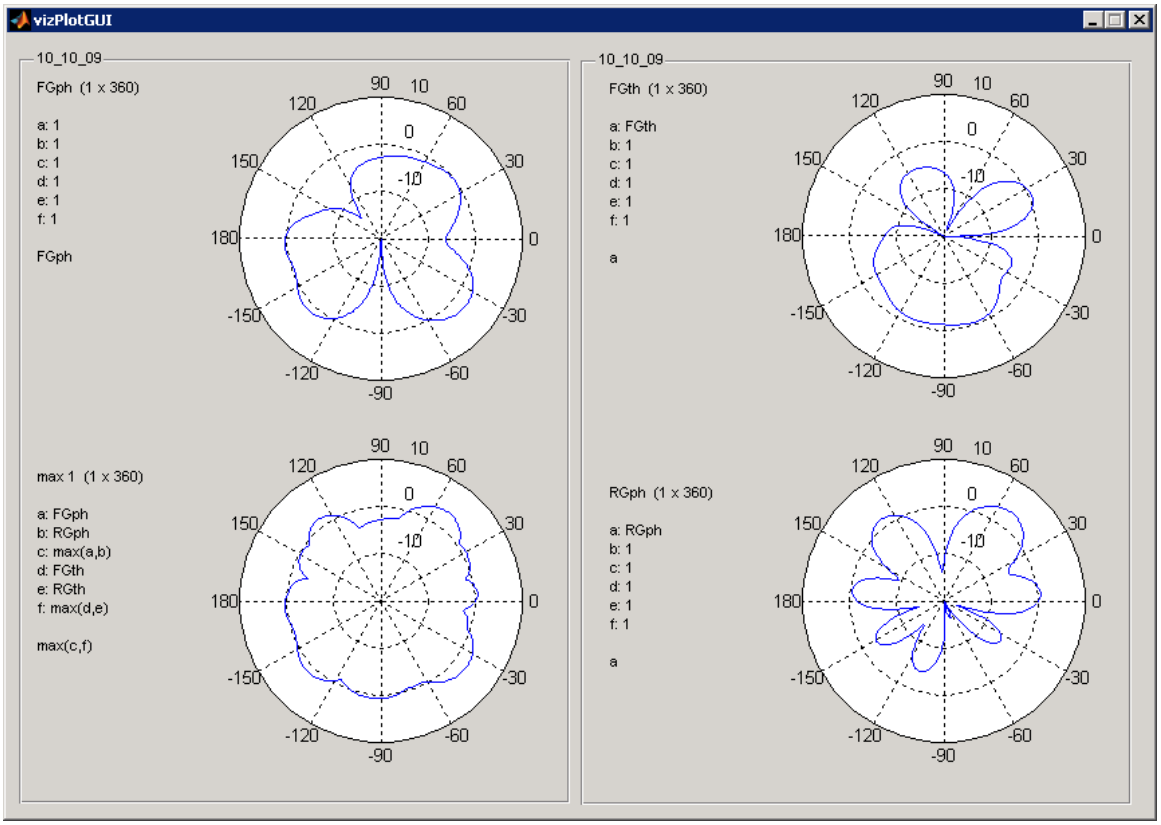


Figure 12: vizPlotGUI, no parameters checked

The visualization GUIs are typically used in the following sequence:

- 3.1 Load Data Files
- 3.2 Define Visualization Functions
- 3.3 Assign Functions to Axes
- 3.4 Choose Parameter Values
- 3.5 Save
- 3.6 Generate a Report

The visualization GUIs allow for the simultaneous display of data from two data sets. The data may be the result of simulation or actual measurements, although the function for

converting measured data to the data format used for simulation results has not yet been implemented (see section 8. Future Improvements 19, 20). This allows for the direct comparison of the theoretical results of two variations of antenna geometry or an antenna's theoretical radiation pattern with the measured radiation pattern of the same geometry. Alternatively, the same data set may be loaded twice for four simultaneous plots of the same data. Note that modifications to default line weight and fonts in MATLAB may cause some elements of the GUI to display incorrectly (see section 8. Future Improvements **Error! Reference source not found.**).

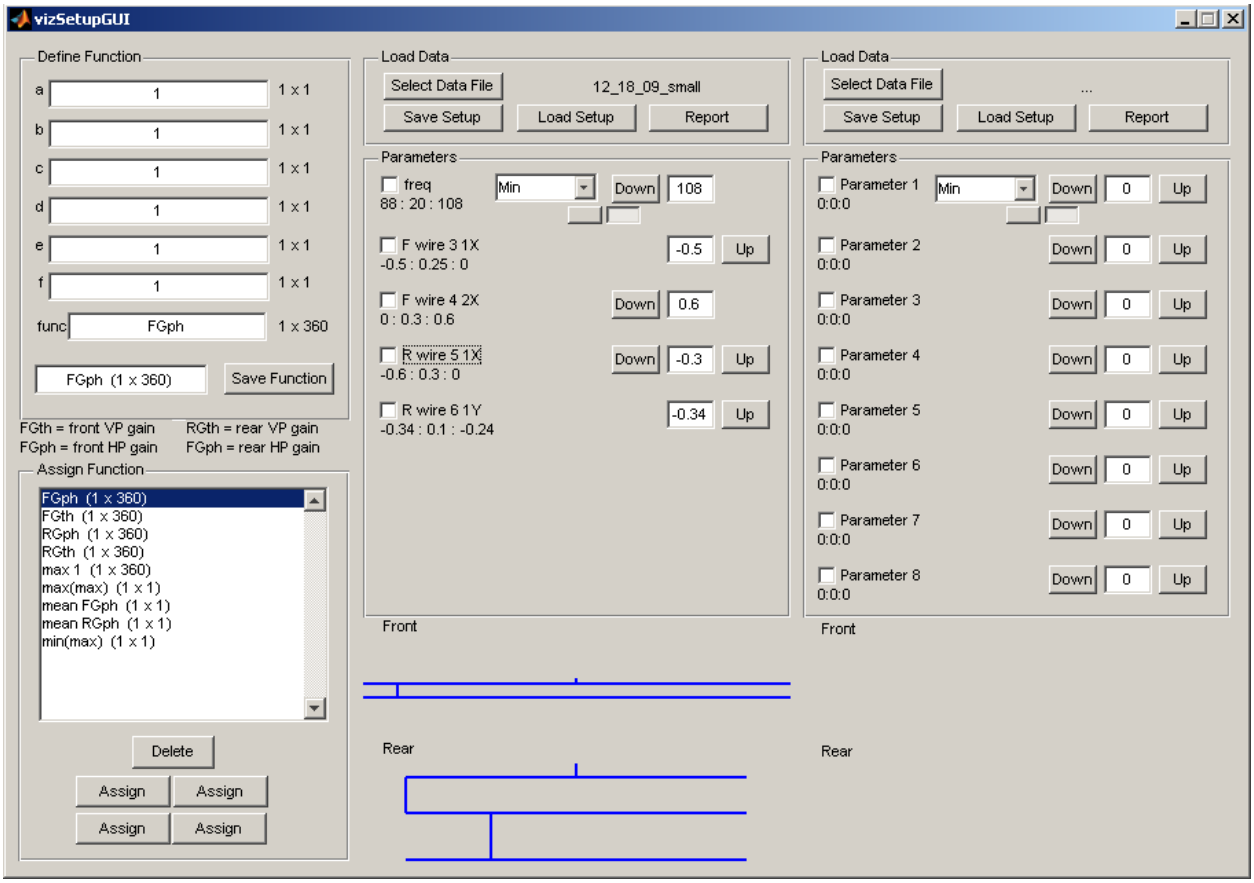


Figure 13: vizSetupGUI With Loaded Data

3.1 Load Data Files

To load a data set, click the **Select Data File** button in the Load Data panel, as show in Figure 14, and select the data file. Currently, data files are expected to have the file extension *.data. Files with other extensions may be used, but they will be filtered out by the file selection window. When browsing for a mesh file with an extension other than *.data, select "Files of Type" → "All files". All data files must share the data structure of *.data files (see section 7. *.data File Data Structure, page 41).

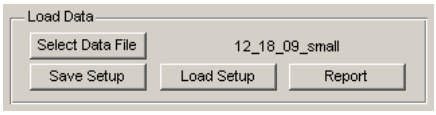


Figure 14: Load Data File

Once a data file has been loaded, **vizSetupGui** will identify and display the swept parameters in the **Parameters** panel. Figure 15 shows an example swept parameter.

Figure 15: Swept Parameter

A description of the parameter will be displayed on the left. In the case of Figure 15, the parameter, given as **R wire 5 1X,** is the x-coordinate of the first endpoint of the fifth wire of the Rear antenna. Below the parameter description the swept values are displayed. In the case of Figure 15, the parameter was swept from -0.6m to 0m with an increment of 0.3m. When first loaded, the data point corresponding to the minimum value of all swept parameters will be

selected. Other parameter values may be selected by pressing the **Up** and **Down** buttons (see section 8. Future Improvements 21). The selected parameter value will affect the display of antenna geometry and the data that is plotted in **vizPlotGUI**.

When a *.data file has been loaded, a plot of the antenna geometries, shown in Figure 16, is displayed below the **Parameters** panel.

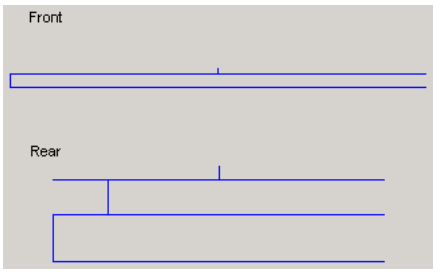


Figure 16: Antenna Plot

3.2 Define Visualization Functions

Visualization functions dictate how data will be displayed in **vizPlotGUI**. All data points in the *.data file contain four values: **FGth** (the vertically polarized gain of the front antenna), **FGph** (the horizontally polarized gain of the front antenna), **RGth** (the vertically polarized gain of the rear antenna), and **FGph** (the horizontally polarized gain of the rear antenna). These values are the direct results of ESP5 simulation and are in units of dBi. Six variables, named **a**…**f** can be defined using the four data values, standard MATLAB functions and standard MATLAB syntax. Variables that are not used must be assigned the number 1 (see section 8. Future Improvements23). Additionally, any variable can access any previous variable. Using Figure 17 as an example, **FGph**, **RGph**, **FGth**, and **RGth** are assigned to variables **a**, **b**, **d**, and **e**, respectively. **c** is defined as the point-by-point maximum of **a** and **b**, and **f** is defined as the

point-by-point maximum of **d** and **e**. The final function is defined as the point-by-point maximum of **c** and **f**. Functions must return either a 1x360 array (which is the size of **FGph**, **RGph**, **FGth**, and **RGth**) or a 1x1 scalar.

Once a function has been defined, a function name is entered in the text box to the left of the **Save Function** button, which is clicked to save the function. It is strongly recommended that the size of the array returned by the function be included in the function name, such as "**max 1 (1 x 360)"**.

Functions can be used to explore the radiation pattern of antennas and also predict the effect of cost functions used with optimization algorithms. A parameter sweep may be analyzed with various functions to find a desirable neighborhood for local optimization or to identify situations where global optimization methods may be appropriate (see section 8. Future Improvements 32).

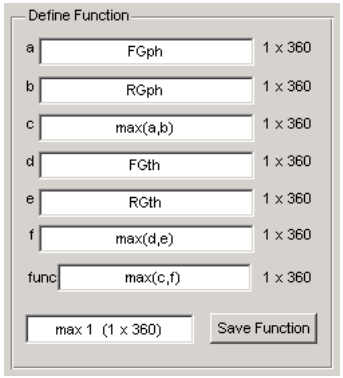


Figure 17: Function Definition

3.3 Assign Functions to Axes

When a function has been saved, its name appears in the list box in the **Assign Function** panel. To delete an unwanted function, select it and click the **Delete** button. The four **Assign** buttons correspond to the four axes in **vizPlotGUI**. When an **Assign** button is pressed, the

corresponding axis will display a plot of the selected visualization function using the data point selected in the **Parameters** panel. The plot is polar, with the radial axis representing azimuthal angle in degrees, and the magnitude of the plot corresponding to the function value at that angle (see section 8. Future Improvements 22, 24). The two axes on the left plot data from the *.data file selected on the left of **vizSetupGUI**, and the two axes on the right plot data from the *.data file selected on the right. The function variables and final function definition are displayed to the left of the axes. The data file name is displayed as the title of the axis panel.

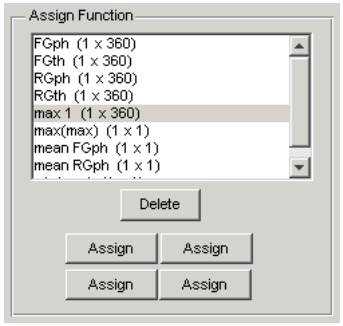


Figure 18: Function Assignment

3.4 Choose Parameter Values

The antenna geometry and data plots correspond to the parameter values selected in the **Parameters** panel. Incrementing parameter values with the **Up** and **Down** buttons will cause dynamic updates of the plots, which will display the antenna geometries and function values associated with the selected parameters.

It is often desirable to visualize in a single plot how an antenna radiation pattern varies across a swept parameter. This is accomplished by checking the checkbox to the left of a parameter in the **Parameters** panel, as shown in Figure 19.

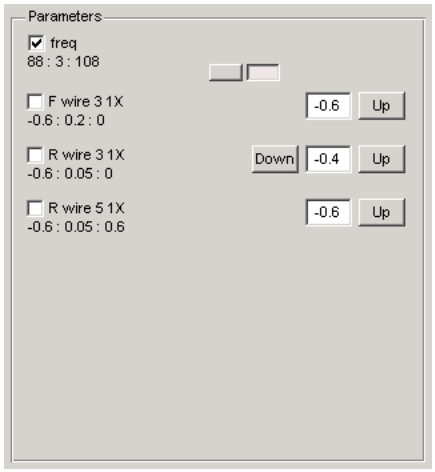


Figure 19: One Parameter Checked

When one parameter has been checked the axes in **vizPlotGUI** will change to orthagonal plots that show function values with azimuthal angle on the x-axis and the checked parameter on the y-axis, as shown in Figure 20. When a parameter is checked, the antenna geometry plots will be hidden, as it is currently not possible to plot an antenna with multiple geometry configurations (see section 8. Future Improvements 25, 26 and 27). The **Up** and **Down** buttons and the parameter value text box associated with the checked parameter are hidden, as all parameter values are being used simultaneously.

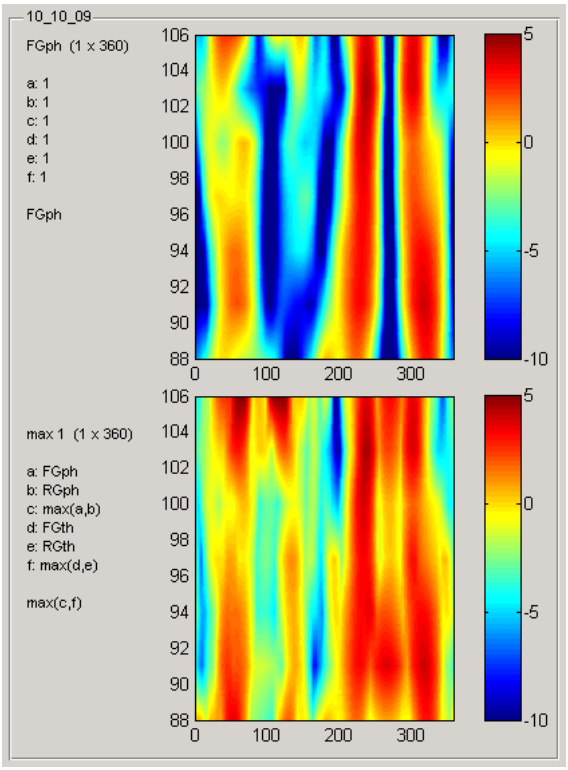


Figure 20: vizPlotGUI, One Parameter Checked

To visualize how two swept parameters vary simultaneously, a second parameter may be checked in the **Parameters** panel, as shown in Figure 21.

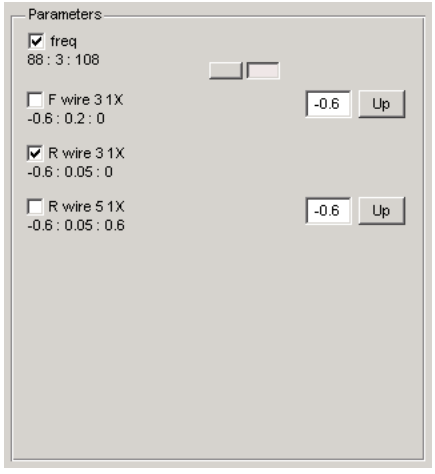


Figure 21: Two Parameters Checked

When two parameters are checked, the axes will display the first checked parameter (as counted from the top down) on the x-axis, and the second parameter on the y-axis (See section 8. Future Improvements 28). Because the plot is no longer a function of azimuthal angle, the assigned visualization functions must return a 1x1 scalar instead of a 1x360 array.

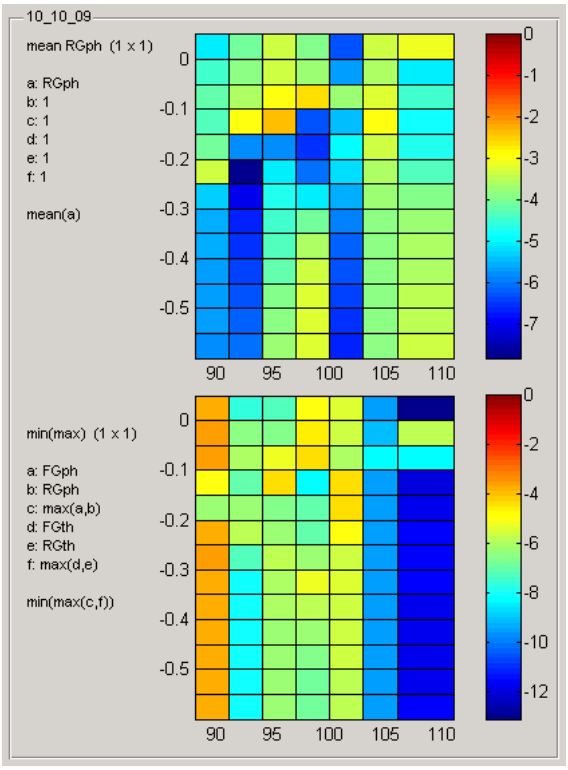


Figure 22: vizPlotGUI, Two Parameters Checked

The following function has yet to be implemented (see section 8. Future Improvements 29). When the **freq** parameter in the **Parameters** panel is not checked, a pair of toggle buttons are shown next to the **freq** parameter, as shown in Figure 23. When the right button is selected, the frequency parameter behaves like all other swept parameters. When the left button is selected, the **Up** and **Down** buttons and the parameter value text box are hidden and a pull-down menu appears. The operation selected from the pull-down menu dictates which frequency value

will be used when calculating visualization function values. As an example, if the frequency pull-down menu is set to **Min** and the selected visualization function simply returns **FGth**, the function value will be evaluated at every swept frequency, and the function returns a 1x360 array of the smallest value of **FGth** at each azimuthal degree. This makes it easy to identify the worst-case radiation pattern of antennas across a frequency band.

Figure 23: Toggle Frequency Options

3.5 Save Setup

Visualization settings may be named and saved at any time. The **Save Setup** and **Load Setup** buttons, shown in Figure 14, are used to save and load the parameter and plot settings of the relevant column of **vizSetupGUI**. Note that when a setup is loaded, the name of the data file and not the name of the setup appears in the load data panel.

3.6 Generate a Report

This function has yet to be implemented (see section 8. Future Improvements 30).

4. m-files

4.1 GUI m-files

```
modelGUI.m              - Generates simulation sweeps.
vizSetupGUI.m           - Loads data, sends plots to vizPlotGUI.
vizPlotGUI.m            - Displays plotted data from vizSetupGUI.
```

4.2 Dependent m-files

Setup

```
allcomb.m               - Returns all combinations of input arrays.
                          (c) Jos van der Geest
antennaData2Wires.m     - Creates a matrix of antenna endpoints.
antennaWires2Mesh.m     - Converts a matrix of antenna endpoints to a list of
                          non-intersecting wires.
emailDone.m             - Sends an email when a simulation is finished.
getNumberOfCombos.m     - Returns the number of parameter combinations in a
                          sweep.
makeAntennaMeshSet.m    - Returns an array of all antenna geometries defined
                          by the antenna definition tables.
makeCarMesh. m          - Creates an automobile wire mesh[not used in normal
                          operation].
makeESPFile.m           - Creates an .esp file that ESP5 will use for
                          simulation.
makeSweep.m             - Generates a simulation sweep.
manageModelGUItable.m   - Adds 'xxxx' to unnecessary cells of the modelGUI.
MOV_ESP5.m              - ESP5 MOV move command.
MVX_ESP5.m              - ESP5 MVX close move command.
PLC_ESP5.m              - ESP5 PLC plate coordinates command.
plotAntenna.m           - Plots antenna wires in 2D.
plotAntenna3D.m         - Plots antenna wires in 3D.
plotCar.m               - Plots car and antenna geometry in modelGUI.
RAD_ESP5.m              - ESP5 RAD radiation scan command.
readPlt.m               - Extracts data from "[…]_pat.plt," which is the file
                          that is created by ESP5.
```

```
REM_ESP5.m              - ESP5 REM command.
runESP.m                - Executes an ESP5 simulation.
showCounts.m            - Displays number of parameter combinations and
                          estimated time.
showModelReport.m       - Prints all GUI values to screen.
showSize.m              - displays the size of the matrix that results from
                          the evaluation of a visualization function.
WRC_ESP5.m              - ESP5 WRC command for wire segment.
WRG_ESP5.m              - ESP5 WRG wire generator command.
WRR_ESP5.m              - ESP5 WRR wire radius and conductivity command.
```

Visualization

```
assignFunction.m        - Plots the selected visualization function to the
                          selected axis in vizPlotGUI.
fnAndChecksMatch.m      - Verifies that the selected function is appropriate
                          for plotting.
getFunVal.m             - Evaluates the selected function at the selected
                          point.
GUIvisibility.m         - Changes the visibility of checkboxes and buttons as
                          needed.
handle2param.m          - Returns array of the handle information of elements
                          of vizSetupGui.
loadSavedFunctionNames.m - Populates the visualization function list box.
loadSavedFunctions.m    - Populates the visualization function variable text
                          boxes.
polarLabels.m           - Generates a labeled polar plot.
                          (c)Brian Kat
resetViz.m              - Returns vizSetupGUI and vizPlotGUI to their original
                          state
saveFunction.m          - Adds a new visualization function to the function
                          data file.
showVizReport.m         - Prints all GUI values to screen.
vizGUInewDataL.m        - Prepares data for use by the GUI.
vizGUInewDataR.m        - Prepares data for use by the GUI.
vizPlot.m               - Main function for plotting in vizPlotGUI.
vizStepParam.m          - Increments or decrements parameters.
```

5. Vehicle Wire Mesh Geometry Files

Vehicle wire mesh geometry files are expected to have the file extension *.car. Files with other extensions may be used, but they will be filtered out by the file selection window. When browsing for a mesh file with an extension other than *.car, select Files of Type → All files.

Vehicle wire mesh geometry files must be MATLAB formatted (MAT-file).

Vehicle wire mesh geometry files contain an *n* x 3 x 2 double-precision variable called `carMesh`, where *n* is the number of wires in the wire mesh. `carMesh` is a simple list of the endpoints of all the wires that define the vehicle mesh:

`carMesh(:,:,1) =`

X1,1	Y1,1	Z1,1
X1,2	Y1,2	Z1,2
…	…	…
X1,*n*	Y1,*n*	Z1,*n*

`carMesh(:,:,2) =`

X2,1	Y2,1	Z2,1
X2,2	Y2,2	Z2,2
…	…	…
X2,*n*	Y2,*n*	Z2,*n*

where X1,1 is the x-coordinate of the beginning of the first wire, X2,1 is the x-coordinate of the end of the first wire, etc.

The origin of the coordinate system should be located on the bottom surface of the vehicle, with the bottom of the vehicle in the X-Y plane, with the top of the vehicle in the +Z direction, and the vehicle bilaterally symmetrical about the Y-axis. This allows for proper functioning of the **height off ground** variable, which defines the distance between the vehicle and a mirror image of the vehicle. The mirror image is used to simulate a PEC ground plane (see section

2.4 Set Simulation Parameters, page 21). The location of the origin of the coordinate system in the X-Y plane is arbitrary and can be defined in any way. It is, however, necessary to have a firm understanding of the coordinate system in order to properly map the local antenna coordinate systems onto the coordinate system of the vehicle (see "Locate the antennas on the vehicle").

Although antennas may be defined with overlapping and intersecting wires, the vehicle wire mesh must be suitable for ESP5 simulation, and wires must therefore only contact other wires at the endpoints.

The function makeCarMesh(), in makeCarMesh.m , can be used to create a simple vehicle model as shown in Figure 24. makeCarMesh() is not called by any other function in this software, and thus must be called manually to create a vehicle wire mesh file. makeCarMesh() uses 23 parameters to define the geometry of a vehicle (including a heater grid). The parameters are illustrated in Figure 25, Figure 26, and Figure 27.

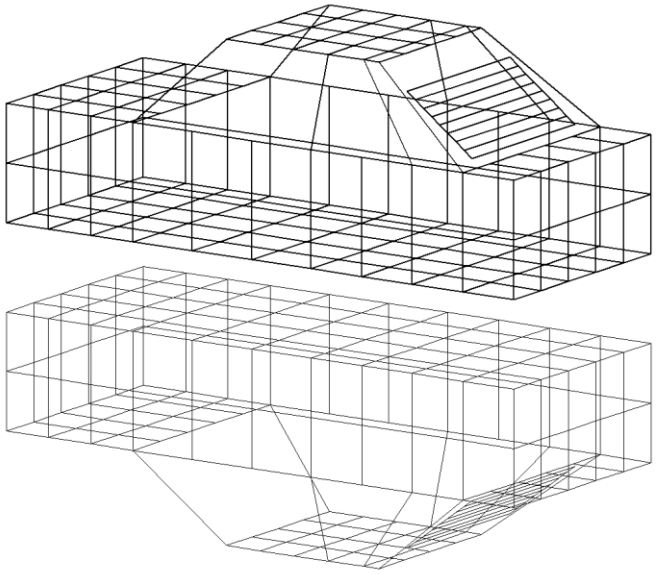


Figure 24: Isometric view of vehicle and image created by makeCarMesh()

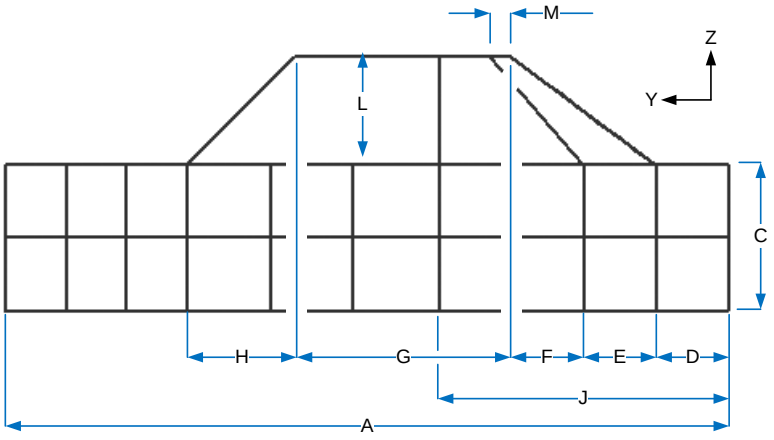


Figure 25: makeCarMesh() parameters, side view

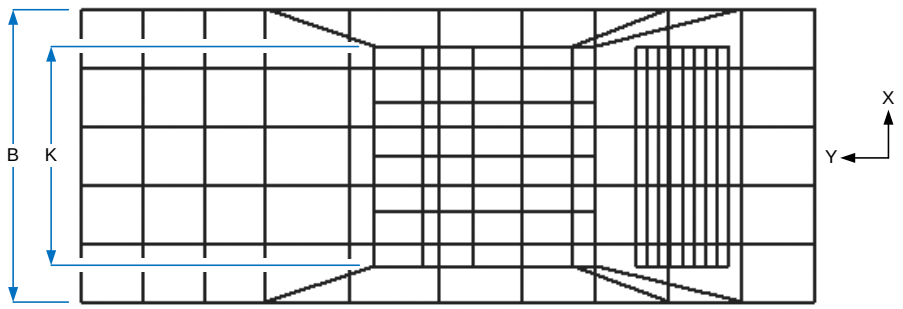
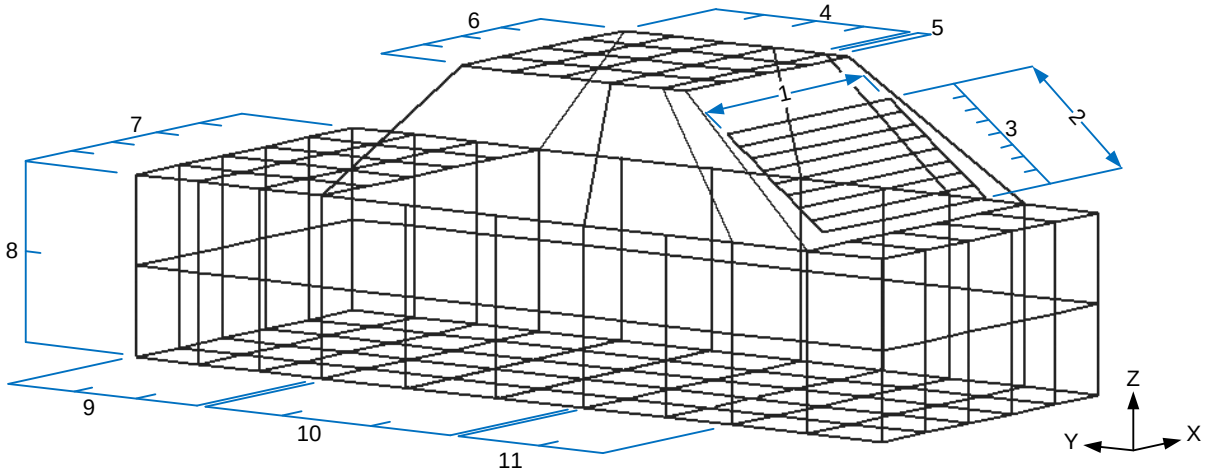


Figure 26: makeCarMesh() parameters, top view



1	heaterWidth	7	meshDensityBodyX
2	heaterHeight	8	meshDensityBodyZ
3	heaterNumberWires	9	meshDensityBodyYa
4	meshDensityRoofYa	10	meshDensityBodyYb
5	meshDensityRoofYb	11	meshDensityBodyYc
6	meshDensityRoofX		

Figure 27: makeCarMesh() parameters, isometric view

The parameters `meshDensityRoofX` and `meshDensityBodyX` must be even numbers due to the construction of the vehicle mesh.

6. Email Notification

The m-file `emailDone()` is called to send an email notification when the simulation is completed.

These variables must be directly entered into the file:

`mail` is the email address to which the messages will be sent
`password` is the password to the email account. [Caution: anybody can see these]
`subject` is the subject line of the email
`body` is the text of the email

The functionality of `emailDone()` could be expanded to send periodic updates, such as "Parameter combination 50 of 1920 completed," or error notifications, such as "ESP5 exited with error code 0."

7. *.data File Data Structure

The results of a sweep are saved in a data file with the file extension `*.data`. Unlike many data file formats (such as comma separated variable files) these files contain a saved MATLAB variable, `sweepData`, and therefore can only be generated or read by MATLAB. `sweepData` is a 1x1 structure with sixteen fields. Using the sweep shown in Figure 28 as an example, we will examine each of the fields of `sweepData`. To manually save a `*.data` file, such as when converting measured data to a format that can be read by **vizSetupGUI**, use the command `save('filename.data','sweepData')`. `*.data` files can be loaded into the MATLAB workspace for editing using the command `load filename.data -mat`.

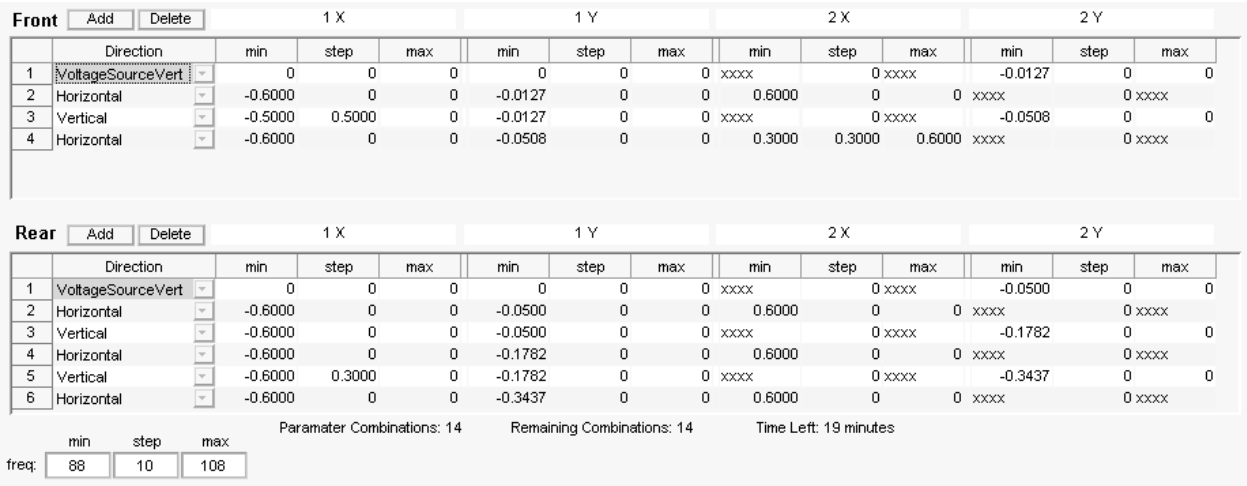


Figure 28: Sweep Setup Example

In the discussions that follow, n is the total number of swept parameters, n_1 and n_2 are the number of parameters in the antennas designated Front and Rear, respectively, including frequency. In the case of Figure 28, $n = 4$, $n_1 = 3$, and $n_2 = 2$.

`sweepData.min: [88 -0.5000 0.3000 -0.6000]`

`sweepData.min` is a 1x*n* array of type `double`. It contains the minimum values of all swept parameters.

`sweepData.step: [2 0.5000 0.3000 0.3000]`

`sweepData.step` is a 1x*n* array of type `double`. It contains the step values of all swept parameters.

`sweepData.max: [108 0 0.6000 0]`

`sweepData.max` is a 1x*n* array of type `double`. It contains the maximum values of all swept parameters.

`sweepData.name: {'freq' 'F wire 3 1X' 'F wire 4 2X' 'R wire 5 1X'}`

`sweepData.name` is a 1x*n* cell array. Each cell contains a character array with the name of the swept parameter.

`sweepData.pValues: {[88 98 108] [-0.50 0] [0.30 0.60] [-0.60 -0.30 0]}`

`sweepData.pValues` is a 1x*n* cell array. The n^{th} cell contains all values of the n^{th} parameter. Although `sweepData.pValues` is redundant with `min`, `step`, and `max`, it is assumed that pre-calculating these values increases the speed of plotting data.

`sweepData.pIndex: {[1 2] [1 2] [1 2] [1 2 3]}`

`sweepData.pIndex` is a 1x*n* cell array. Each cell contains a simple index of the swept parameters. Note that cell `i` of `sweepData.pIndex` could be obtained by `1:size(sweepData.pValues{i},2)`. As with `sweepData.pValues` itself, it is assumed that pre-calculating `sweepData.pIndex` will result in computational efficiency.

`sweepData.param: {[4-D double] [2x3x2 double]}`

`sweepData.param` is a 1x2 cell array that is intended to easily access selected parameter values for visualization function evaluation and plotting. The first and second cells correspond to the antennas designated Front and Rear, respectively. The first cell contains a n_1+1 dimensional array, and the second cell contains a n_2+1 dimensional array. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `param{1}(`i_1`, `i_2`, … ,`i_n`,:)` returns the

parameter values associated with the parameter selection. Because the frequency parameter applies to both Front and Rear antennas, frequency will be the first parameter for both `sweepData.param{1}` and `sweepData.param{2}`. For example, if data file that results from the sweep shown in Figure 28 is loaded into **vizGUI** and the first parameter's **up** button is pressed twice, the third parameter's **up** button is pressed once, and the fourth parameter's **up** button is pressed once, `sweepData.param{1}(3,1,2,:)` will return the values `108`, `-0.5`, and `0.6`. `sweepData.param{2}(3,2,:)` will return the values `108` and `-0.3`.

`sweepData.antennaMesh: {[2x2x2 cell]  [2x3 cell]}`

`sweepData.antennaMesh` is a 1x2 cell array containing antenna mesh data for both antennas. The first and second cells correspond to the antennas designated Front and Rear, respectively. Each of these cells contains another cell array. These cell array within the first cell is n_1 dimensional, and the cell array within the second cell is n_2 dimensional. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `sweepData.antennaMesh{1}(`i_1`,` i_2`, … ,`i_n`)` returns the antenna corresponding to the selected parameters. Note that antenna geometries are stored for each frequency, which leads to redundant information. The redundancy is tolerated so that accessing antenna meshes is as intuitive as possible.

`sweepData.theta: [360x1 double]`

`sweepData.theta` contains the values of theta used for the sweep. Currently, the values will be all be the value defined in **modelGUI**.

`sweepData.phi: [360x1 double]`

`sweepData.phi` contains the values of phi used for the sweep. It will usually contain `1:1:360`.

`sweepData.gainTheta: {[4-D double]  [2x3x360 double]}`

`sweepData.gainTheta` is a 1x2 cell array containing the vertically polarized gain for each parameter combination. The first and second cells correspond to the antennas designated Front and Rear, respectively. The first cell contains a n_1+1 dimensional array, and the second cell contains a n_2+1 dimensional array. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `sweepData.gainTheta{1}(`i_1`,` i_2`, … ,`i_n`,:)` returns a 360-element array, which is the vertically polarized gain of the Front antenna at the selected parameters across all 360 azimuthal degrees. `sweepData.gainTheta` is in units of dBi.

`sweepData.gainPhi: {[4-D double]  [2x3x360 double]}`

`sweepData.gainPhi` is a 1x2 cell array containing the horizontally polarized gain for each parameter combination. The first and second cells correspond to the antennas designated Front and Rear, respectively. The first cell contains a n_1+1 dimensional array, and the second cell contains a n_2+1 dimensional array. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `sweepData.gainPhi{1}(`i_1`, `i_2`, … ,`i_n`,:)` returns a 360-element array, which is the horizontally polarized gain of the Front antenna at the selected parameters across all 360 azimuthal degrees. `sweepData.gainPhi` is in units of dBi.

`sweepData.angTheta: {[4-D double]  [2x3x360 double]}`

`sweepData.angTheta` is a 1x2 cell array containing the vertically polarized phase angle for each parameter combination. The first and second cells correspond to the antennas designated Front and Rear, respectively. The first cell contains a n_1+1 dimensional array, and the second cell contains a n_2+1 dimensional array. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `sweepData.angTheta{1}(`i_1`, `i_2`, … ,`i_n`,:)` returns a 360-element array, which is the vertically polarized phase angle of the Front antenna at the selected parameters across all 360 azimuthal degrees. `sweepData.angTheta` is in units of degrees.

`sweepData.angPhi: {[4-D double]  [2x3x360 double]}`

`sweepData.angPhi` is a 1x2 cell array containing the horizontally polarized phase angle for each parameter combination. The first and second cells correspond to the antennas designated Front and Rear, respectively. The first cell contains a n_1+1 dimensional array, and the second cell contains a n_2+1 dimensional array. If the i^{th} value of each parameter of the Front antenna is selected in **vizGUI**, then `sweepData.angPhi{1}(`i_1`, `i_2`, … ,`i_n`,:)` returns a 360-element array, which is the horizontally polarized phase angle of the Front antenna at the selected parameters across all 360 azimuthal degrees. `sweepData.angPhi` is in units of degrees.

`sweepData.gainThetaShift: {[4-D double]  [2x3x360 double]}`

`sweepData.gainThetaShift` contains the same information as `sweepData.gainTheta` , but shifted +20dB with a floor of 0dB. `sweepData.gainThetaShift` is used for polar plotting.

`sweepData.gainPhiShift: {[4-D double]  [2x3x360 double]}`

`sweepData.gainPhiShift` contains the same information as `sweepData.gainPhi` , but shifted +20dB with a floor of 0dB. `sweepData.gainPhiShift` is used for polar plotting.

8. Future Improvements

8.1 Setup

1. Allow user to create only one antenna. Currently the software requires two antennas to be defined, although it is possible to create a simple antenna without any swept parameters, resulting in "wasted" simulations equal to the number of frequencies used in the sweep.
2. Allow user to have an antenna with static geometry. Currently, the software requires at least one swept parameter.
3. Allow antennas to be defined in three dimensions. Currently, antennas are defined on a flat, planar surface.
4. Add rounding to `antennaData2Wires`. This is to avoid the creation of very small wires that will cause an error in ESP5.
5. Force the first wire of an antenna to have a voltage source. Only allow simple horizontal or vertical options for all subsequent wires. It is currently necessary for the voltage source to be on the first wire for the algorithm that splits the initial set of antenna wires into a set of non-overlapping wires to function properly (see `antennaWires2Mesh.m`).
6. Generally improve antenna definition. Right now it's not very intuitive.
7. Improve checks for very short wires (that will be rounded to zero and cause ESP5 to crash). Currently, there is rounding in `antennaData2Wires.m`, but more work may be required to make the antenna creation algorithms sufficiently robust.
8. Improve estimated time remaining by accounting for longer simulation times at higher frequencies.

9. Show all locations of swept antenna wires. Display swept wires in a different color than the rest of the antenna.
10. Allow for antennas to be rotated about the other two axes. This will allow more flexibility in the location of antennas (side windows, etc.)
11. Use degrees for defining antenna angle theta. This is for consistency with the angle theta that defines the altitude of the simulation.
12. Label X, Y, and Z axes on vehicle plot.
13. Save changes to antenna location as soon as they have been made, without requiring a change to the antenna definition table.
14. Allow altitude of simulation (theta) to be swept.
15. Generate a default simulation name using the current date.
16. Add settings for an email to be sent indication progress or error.
17. Incorporate additional simulation engines, particularly ones that are more efficient with higher frequencies.
18. Modify the output of text file reports to fit onto standard paper sizes.

8.2 Visualization

19. Implement algorithm to convert measured radiation pattern data to simulated radiation pattern format.
20. Allow viz functions to use data from both loaded data sets.
21. Allow the user to manually enter parameter values in text box.
22. Use list dialog listdlg() instead of listbox to display saved functions.
23. Allow function variables to be blank if they are not used (currently need to be 1).
24. Add dBi label to all plots? Functions may change units…

25. Add labels to axes when one or more parameters are checked.
26. Allow user to control range of pcolor plots.
27. Plot all antenna geometry configurations simultaneously when parameters are checked. Draw swept wires in a distinct color.
28. When two parameters are checked, allow user to select which parameter is on which axis (x-axis or y-axis).
29. Implement frequency toggle buttons.
30. Add Generate Report function to vizSetupGUI.
31. Simulate antenna signal multiplexing.

8.3 General

32. Add tools for optimization in a new GUI.
33. Improve file/folder management and MATLAB search paths.
34. Allow GUIs to be used when the current MATLAB directory has been changed.
35. Override setup modifications to default line weight and fonts.

9. References

[1] Blank, S. J., & Hutt, M. F. (2008, April 28-30). Antenna array synthesis using derivative, non-derivative and random search optimization. *Sarnoff Symposium, 2008 IEEE* , 1-4.

Test problems are used to compare the convergence rates of derivative (NLFI), non-derivative direct search (Nelder-Mead simplex and finite difference quasi-Newton), and random search (particle swarm) optimization techniques. It is shown that the derivative based methods are more efficient than the non-derivative direct search methods without randomness, which in turn are more efficient than direct search methods with randomness. These are expected results. The non-derivative direct search techniques are generally less efficient than derivative techniques, but are useful when gradient information is unavailable. Random search techniques converge very slowly, but have better exploratory characteristics, and are useful for finding global minima.

[2] Bregon, C. D., & del Rio, E. F. (2002). Hybrid optimizer based on genetic algorithms and conjugate gradient. *Antennas and Propagation Society International Symposium, 2002, IEEE , 1,* 738-741.

The authors present commercial optimization software that implements a hybrid genetic algorithm and gradient-based approach. It was designed for electromagnetics problems, but is broadly applicable. The software begins by using the genetic algorithm to find the neighborhood of a global solution. The gradient-based method is used to improve the accuracy of the solution found by the genetic algorithm. This approach combines the exploratory power of the genetic algorithm with the rapid convergence of gradient-based methods.

[3] Csiszar, S. (2007). Optimization algorithms (survey and analysis). *International Symposium on Logistics and Industrial Informatics,* (pp. 185-188). Wildau.

The author first surveys optimization algorithms. Classical heuristics are distinguished from

metaheuristics. Within metaheuristics, a large group called Adaptive Memory Programming (AMP) related heuristics is identified, including tabu search, evolutionary and genetic algorithms, scatter search, and ant colony optimization. Simulated annealing is located outside of the AMP related techniques because it is memoryless. It is noted that simulated annealing, in certain circumstances, can be guaranteed to find the optimum solution. The navigation characteristics and memory structure of several algorithms are compared. This paper focuses on how these algorithms work, and does not provide much practical information that would inform a decision between the algorithms.

[4] Gandelli, A., Grimaccia, F., Mussetta, M., Pirinoli, P., & Zich, E. (2007). Genetical swarm optimization: self-adaptive hybrid evolutionary algorithm for electromagnetics. ***Antennas and Propogation, IEEE Transactions on , 55*** **(3), 781-785.**

A hybrid genetic algorithm/particle swarm optimization technique is proposed. The design of a printed reflectarray antenna is optimized with genetical swarm optimization, genetic algorithm and particle swarm optimization techniques. It is shown that genetical swarm optimization performed best for this design problem.

[5] Guney, K., & Onay, M. (2008). Bees algorithm for design of dual-beam linear antenna arrays with digital attenuators and digital shifters. ***International Journal of RF and Microwave Computer-Aided Engineering, 18*** **(4), 337-347.**

The authors demonstrate the application of a bees algorithm to a design problem. It is noted that the bees algorithm is very accurate and does not require complicated mathematical functions.

[6] Haupt, R. (1995). Comparison between genetic and gradient-based optimization algorithms for solving electromagnetics problems. ***Magnetics, IEEE Transactions on , 31*** **(3), 1932-1935.**

The authors use several electromagnetic optimization problems to compare a gradient-

based algorithm (the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm) with a standard genetic algorithm. It is discovered that the gradient-based algorithms perform best for a small number of continuous parameters and genetic algorithms perform best for a large number of quantized parameters. It is noted that genetic algorithms can be used to find good starting points for continuous optimization algorithms.

[7] Kim, Y., & Walton, E. K. (2006). Automobile conformal antenna design using non-dominated sorting genetic algorithm (NSGA). ***Microwaves, Antennas and Propagation, IEE Proceedings, 153*** **(6), 579-582.**

The authors apply a non-dominated sorting genetic algorithm to the problem of optimizing conformal automobile antennas. This algorithm differs from simple genetic techniques in that it can compromise between multiple design objectives. It is shown that algorithm can successfully modify an existing antenna design to compromise between gain and VSWR. The algorithm is not compared to any other techniques.

[8] Koo, J. C., Shim, J., Suh, T. I., Bang, K., J., & Kim, H. T. (2004). Size optimization of a microstrip GPS antenna for automobile glass with a genetic algorithm. ***Journal of Electromagnetic Waves and Applications , 18*** **(11), 1459-1470.**

A simple genetic algorithm is used to successfully optimize a GPS antenna for automobile glass. The algorithm is not compared to any other techniques.

[9] Lee, K. C. (2002). Optimization of bent wire antennas using genetic algorithms. ***Journal of Electromagnetic Waves and Applications , 16*** **(4), 515-522.**

A simple genetic algorithm is used to optimize the shape of a bent wire antenna. The general advantages of genetic algorithms over traditional gradient-based techniques is discussed, although no other optimization techniques are actually applied to the design problem.

[10] Li, Z., Erdemli, Y. E., Volakis, J. L., & Papalambros, P. Y. (2002). Design optimization of conformal antennas by integrating stochastic algorithms with the hybrid finite-element method. ***IEEE Transactions of Antennas and Propagation , 50*** **(5), 676-684.**

The authors combine non-deterministic optimization algorithms (genetic algorithms and simulated annealing) with finite-element boundary-integral simulations to design and optimize 3-D antenna geometry and material specifications. The paper includes flowcharts that succinctly describe genetic algorithms and simulated annealing.

[11] Misevicius, A., Blazauskas, T., Blonskis, J., & Smolinskas, J. (2004). An overview of some heuristic algorithms for combinatorial optimization problems. ***Information Technology and Control , 30*** **(1), 7-12.**

The authors discuss the general characteristics of several optimization algorithms: descent local search, simulated annealing, tabu search, genetic algorithms, ant algorithms, and iterated local search. The performance of each of these algorithms when applied to the quadratic assignment problem is shown. The genetic algorithm and the iterated local search algorithm performed best.

[12] Olcan, D. I., Golubovic, R. M., & Kolundzija, B. M. (2006, July 9-14). On the efficiency of particle swarm optimizer when applied to antenna optimization. ***Antennas and Propagation Society International Symposium 2006, IEEE ,*** **3297-3300.**

A standard particle swarm optimization algorithm is applied to three antenna optimization problems. Its performance is compared to a gradient algorithm and the Nelder-Mead simplex algorithm. The results demonstrate that particle swarm optimization performs best for optimization problems with a medium number of variables, although it compared well in all problems. It is noted that particle swarm optimization is well suited for implementation on parallel processors.

[13] Cerretelli, M., & Gentili, G. B. (2007). Progress in compact multifunction automotive antennas. Electromagnetics in Advanced Applicaitons, 2007 ICEAA 2007, International Conference on , 93-96.

The authors discuss their progress with vehicle mounted multifunction antennas. Whip and rod antennas, antennas concealed in wing mirrors or bumpers, roof and cabin mounted multi-band antennas, and satellite multifunction antennas are discussed.

[14] Filipovic, D. S., & Volakis, J. L. (n.d.). Multifunctional conformal antennas for automobile applications.

The authors present two new conformal, shallow and cavity-backed, slot spiral antennas with multi-band properties. The antennas are intended for simultaneous reception of the (terrestrial-based) Digital Audio Broadasting and (satellite-based) Satellite Digital Audio Radio Services systems. These antennas are designed to be mounted on an automobile.

[15] Wang, J. J. (n.d.). Conformal multifunction antenna for automobiles.

The author discusses the use of a spiral-mode microstrip antenna as a multifunction automobile antenna. Difficulties providing an adequate body cavity are discussed, as are efforts to reduce the size of the antenna with new antenna designs and materials.

[16] Lindenmeier, S., & Brose, J. (2008). Numerical modeling of car antennas. (P. Russer, & U. Siart, Eds.) Springer Berlin Heidelberg.

The authors discuss several ways of modeling the characteristics of multifunctional antennas on automobiles. Volume discretization, surface discretization, quasi-optical, and hybrid methods are discussed. The advantages of antenna diversity are discussed. The authors present a system for diversity optimization using measured and simulated antenna data.

[17] Kronberger, R., Lindenmeier, H., Hopf, J., & Reiter, L. (1997). Design method for antenna arrays on cars with electrically short elements under incorporation of the radiation properties of the car body. ***Antennas and Propagation Society International Symposium, 1997. IEEE., 1997 Digest*** **, 418-421.**

The authors present a car phone antenna at the upper rim of the rear window of an automobile. The antenna consists of several narrowly spaced short antenna elements. Car bodies tend to interfere with the omnidirectional characteristics of car phone antennas. These problems are usually resolved with a quarter-wave vertical antenna. Multiple short elements avoid these problems without an intrusive vertical antenna. The simplex method was used for optimization, rather than a gradient technique.

[18] Lindenmeier, H., Hopf, J., Reiter, L., & Kronberger, R. (1999). Optimization of the antenna-diversity-effectiveness of complex FM-car-antenna systems. Antennas and Propagation Society International Symposium, 1999. IEEE , 2058-2061.

The authors show that compact antenna systems on a single car window can provide high diversity effectiveness for FM radio. Antennas on multiple windows are shown to be unnecessary.

[19] Abou-Jaoude, Ramzi N. (1997). Design and development of conformal automobile antennas using numerical modeling and experimental techniques. PhD dissertation, The Ohio State University.

The author surveys techniques for the design and development of conformal automobile antennas. Topics relating to numerical modeling are presented, and the benefits of antenna diversity systems, antenna arrays, and multifunctional antennas are presented.

[20] Kim, Yongjim. (2003). Development of automobile antenna design and optimization for FM/GPS/SDARS applications. PhD dissertation, The Ohio State University.

The author addresses the application of multi-objective genetic algorithms to the design of automobile antennas. The design requirements of automotive antennas are discussed. The author presents computational code for the automated optimization of rear-window automobile antennas with Nondominated Sorting Genetic Algorithms.

[21] Villarroel, Wladimiro. (2002). Automated design and optimization of VHF/UHF automotive conformal antennas. PhD dissertation, The Ohio State University.

The author discusses the application of genetic algorithms to the design of conformal FM automobile antennas. Special emphasis is placed on the careful development of a multi-objective cost function for optimization. Several useful data visualization techniques are shown.